Service broker based on cloud service description language

Razaq, Abdul; Tianfield, Huaglory; Barrie, Peter; Yue, Hong

Published in:
15th International Symposium on Parallel and Distributed Computing (ISPDC), 2016

DOI:
10.1109/ISPDC.2016.34

Publication date:
2017

Document Version
Author accepted manuscript

Service Broker for Cloud Service Description Language

Abdul Razaq, Huaglory Tianfield, Peter Barrie
School of Engineering and Built Environment,
Glasgow Caledonian University, Glasgow, UK
{ abdul.razaq, h.tianfield, peter.barrie } @gcu.ac.uk

Hong Yue
Dept. of Electronic and Electrical Engineering
University of Strathclyde, Glasgow, UK
hong.yue@strath.ac.uk

***Abstract*- Cloud Service Description Language (CSDL), initiative and discourse, is concentrated to deploy applications on various cloud platforms without modifying source-code. Semantic topology and orchestration of applications provides practical advantage for service providers with ability of interoperability, portability and unified interfaces. However, this has also resulted problems for consumers to identify the appropriate services spread over swarm platforms. The advantage, with common CSDL such as Topology and Orchestration Specification for Cloud Applications (TOSCA), becomes problematic for consumers. Service providers will have different technical and business details such as: discovery, pricing, licensing or composition depending upon deployed platform; therefore, selection of service becomes challenging and requires human effort. Service Broker design is presented for TOSCA framework only; however, the suggested scheme is generic and adaptable to accommodate similar standards of CSDL.**

***Index Terms*— Cloud Services, TOSCA, Service Management, Service Broker, Cloud Description Language**

I. INTRODUCTION

An independent application, on an isolated machine serving certain functionality, did not require any non-functional description except technical usage details such as data type and size with its timing constrains. Then, evolved distributed architectures running, multiple applications which undoubtedly required more than technical usage including IP, connecting ports and send/receive methods. Now, with these distributed systems evolving into next abstracted world where a single machine can be either visualized as a unit or dispersed devices. Cloud computing has introduced a method,

Fig. 1. Evolution of Application Services Technologies from standalone systems to distributed networks and Cloud platforms.

to virtualize and interoperate, a single application on different systems that are different by definition with comparable execution environment. Evolving applications must adopt virtualization where technical details serve no significance but cost, quality and availability matters.

A service is: “function of input or output originating to or from either hardware or software”. Historically, services have been described through technical details only (Figure 1.) with Remote Procedure Calls (RPC) as beginning of distributed computing era. The business aspect of these services have been ignored and left upon the users to compare, search and select appropriate service that fulfilled the business requirements. Moreover, the technical descriptions have been focused on specific platforms or execution environments only. As a result, the gap between technical and business requirements is focused in on-going research. There is a global push to standardize Cloud Based Web Services that has lead various consortiums including OMG, W3C, ISO, NIST and Eurocloud resulting into CSDL standards such as USDL, CCRA, OCCI and TOSCA [1].

TOSCA enabled services become 1) Marketable Entities that can be listed in catalogues via service templates, 2) it also ensures that IT services defined as Service Templates are portable, 3) And finally instance of a service can be composed from different components provided by different providers and hosted on different architectures.

TOSCA initiative describes a service component with two directives. 1) A service 'Topology' is a directive to describe the service and its relationship with other components. 2) Orchestration provides management description in terms of service creation and modification. This combination allows application providers to deploy across various alternative cloud environments that are compatible with TOSCA containers. TOSCA serves three domains: 1) Design Tools for applications as services 2) Service market places to service brokers that can utilize the provided/designed application and 3) Cloud providers that can manage their resources to host provided application [2] [3] [4].

The idea behind text based description technologies is to provide human readable but machine usable documents to describe offered applications and related services, Cloud platforms have further pushed this direction to allow such technologies to include ways to combine these services and make it deployment independent in terms of underlying platforms. This has resulted into Service Template what serves as blue print to deploy a piece of software across different but similar executions. The major focus lies in management of this transition with minimal overhead of services transformation for targeted execution platform.

The TOSCA description documents refer to applications in terms of Service Topology Templates and Service Plans.

TOSCA grammar provides a mechanism that allows adding new descriptions that are application or deployment specific. Process Execution Language Version (BPEL), Business Process Model and Notation (BPMN), XML Path Language (XPath) and Open Virtualization Format Specification (OVF) are similar initiative as TOSCA but with different governing bodies.

The remainder of the paper is organized as follows. Section 2 reviews the literature on Cloud service description techniques and management. Section 3 presents our proposed TOSCA Service Broker scheme with detailed design and workflow. Section 4 provides directions to future work followed by conclusion in last section.

II. LITERATURE REVIEW

TOACA meta-model consists of two structures that are Topology Template (aka topology model of a service) and Plans. Topology is a way to refer modeling of service as a node with or without association, whereas Plans describe lifecycle of these templates. Life cycle includes creation, runtime, auditing and management. TOSCA relies on BPMN or BPEL to define Plans but any process model language can be used. A template node has interfaces, these are the means to classify the functionally in semantic or ontology in orchestrations. A Topology Template further consists on Node Templates and Relationship Templates to describe the topology model of a service.

Relationship Templates describes semantic and relationship of properties between nodes in a Topology Template. A deployed service is an instance of a Service Template. More precisely, the instance is derived by instantiating the Topology Template of its Service Template, most often by running a special plan defined for the Service Template, often referred as build plan. Service instance is derived from Topology Template of its Service Template according to its build plan. Service templates can be shared in catalogs of service providers.

An artifact in TOSCA is content of executable (database, executable, image, library etc.). These artifacts are further divided into two categories: implementation and deployment. Implementation artifact is actually content and deployment refers to its runtime environment. QoS, service auditing and non-functional behaviors in TOSCA are referred as Policy Template based on Policy Node. A Policy is able to perform auditing, monitoring or payment conditions. A single or set of polices can be attached to a Node template. An archive format CSAR (Cloud Service Archive) is used to encapsulate all the applications that can be deployed to TOSCA container. CSAR file can be considered as compressed TOSCA container description file.

Ghijsen et al. [5] proposed Infrastructure and Network Description Language (INDL) to decouple connectivity, functionality and virtualization of resources with semantic prospective. This modularized approach allows adding new resources without effecting existing resources. Their solution is project specific which present challenges how this can be adopted on large scale.

Baker et al. [6] proposed Intention Description Language (IDL) focused on non-functional requirements of applications. They suggested to modify business models with zero offline elasticity capabilities. This approach lacked standardization and suggested description samples included obsolete rather relative paths.

Binz et al. [7] summarized and analyzed TOSCA containers and how these can expedite application development which can offer services on wider hosting platforms. It has been concluded that widely used echo systems of TOSCA containers is key to broader acceptability. Sun et al. [8] surveyed description techniques for general and basic service scope - SOA with both sides on agreed architecture for system design and virtualization that presents host and client in abstraction. The discussion is further expanded in details such as coverage, representation, users and features. It has been suggested that Unified Service Description Language (USDL) can offer to bridge the gap between technical, operational and fiscal challenges. However, this survey didn't include TOSCA or BEPL4XL.

Cardoso et al. [9] went into great details elaborating gap between technical and business requirements. The focus of their effort is to streamline a description language that can be adopted to run the SAP based products. The emphasis of their effort is based on USDL only and relevant echo-systems. It fails how non SOA based applications can leverage Cloud resources. Similar work conducted by Charfi et al. [10] based on Jorge Cardoes et al. introduced an Eclipsed-based Editor for USDL that enables creation of such models.

Cardoso et al. [1] investigated how USDL and TOSCA can be integrated for management of cloud services. Although, both of these standards serve a common purpose but each has its focused domain in Clouding Computing. USDL is focused on description of services and how they can be used in terms of delivery, discovery and composition. Whereas, TOSCA is more focused on deployment platforms for such services and the requirements of container to host these services. An opensource SugarCRM project has been selected to integrate USDL and TOSCA.

Brogi et at. [11] made a similar effort to summarize the TOCSA with three main prospects (1) automatic deployment and management of applications, (2) portability across different cloud environments, and (O3) interoperability with reusability. Gonçalves et al. [12] proposed XML-based Cloud Modeling Language which can describe distributed resource , services, and requests in an integrated way. The experiments were conducted on virtual environments that might differ when deployed on actual platforms. Moreover, it lacks to describe scalability.

Schaffrath et al. [13] proposed a description language that allows to combine and describe topology and requirements abstracted services. The proposed idea was emulated on a testbed and it further requires validations for different

providers. Silva et al. [14] investigated migration process of services across different platforms. The proposed cloud migration supporting description techniques is at early stages with integration in TOSCA. Cardoso et al. [15] surveyed linked data enabled USDL and its distinctive features. Pedrinaci et al. [16] also investigated linked USDL's vocabulary to support trading services with scalability and automation over multiple cross-domain providers. This work further requires achieving composition of services and a broker mechanism that can provide a common interface for service provider and users.

George et at. [17] proposed description framework based on OWL for publication and discovery mechanism that would allow automated means between providers and brokers. The architecture is based on REST and sematic data source. It requires further optimization and compatibility with other container providers. Lenk et al. [18] investigated conceptual description approach for application run-time requirements in federated clouds. Hoberg et al. [19] investigated service descriptions technologies from customer's perspective. They have identified the information required by service users that can automate search and feature comparison. It has been suggested that USDL can be enhanced to integrate with greater user perspective.

III. PROPOSED ARCHITECTURE

Challenges for Service Consumers:

The initiative of TOSCA standardization has addressed the problems for service providers to deploy the applications on various cloud providers. However, this has also resulted problems for consumers to identify the suitable service. The advantage of TOSCA portability becomes problem for users as different cloud providers will have different technical and business details such as discovery, pricing, licensing or composition; therefore service consumers have to have knowledge of these details.

Figure 2 illustrates this problem with a single TOSCA compatible application capable to be deployed on various cloud providers. Single application can be commissioned to many platforms, type A) platform which is classical machine

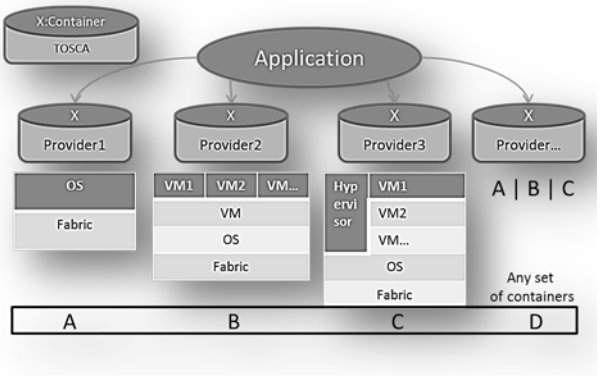

Fig. 2. TOSCA Application deployment on various cloud platforms.

with fabric and operating system, B) underlying cloud platform can be a composition of type 'A' with addition of Virtual Machine Manager (VM Manager) which would allow installing guest operating systems. Another C) type of cloud can be constructed with hypervisor model which allows sharing kernel space for different users, hypervisor is typical use case of different users on single machine with unique identity. Finally, a cloud provider can select a mixture of these topologies D) to offer cloud platform.

On one hand TOSCA provides automation for service providers but on other hand it creates challenges for service consumers. The anonymity of platforms and installed software present challenges for consumer as 1) technical details such as operating system, available applications for given platform, network bandwidth or discovery and composition mechanism and 2) business details such as pricing, licensing, ethical policies or geographical constrains can vary from platform to platform. Another problem is selecting the right service as multiple services can exist for same functionality which presents a selection challenge for consumers.

Some of the challenges are inherited by the TOSCA itself as it does not specify the hardware type (i.g. CPU virtualization, availability of GPU), network bandwidth and typical WSDL problems as it relies on this standard for technical details. On business side, the burden is left on consumer to identify the right service.

Proposed Workflow:

We propose a broker that serves as middleware between service provider and consumer to address the challenges described above. TOSCA Service Broker (TSB) digests the TOSCA description document and produces a leaflet that is an entry in services' catalogue. Aggregated catalogue provide a detailed selection capability to users to select the precise service. A Broker would be able to weight similar services based on their technical and business details.

Figure 3. presents the work flow of proposed TSB. TSB acts as a middleware and gathers all TOSCA documents from different service provider regardless of underlying cloud platform. All TOSCA documents are parsed and information is stored in database for further presentation and analytics. Users interact with TSB for service selection and are served with a catalogue of services based on criteria and best match. Users' applications can be automated to select the right service as soon as new competent services are added in catalogue. In essence, TSB collects applications' description and arranges these details in catalogue format. Furthermore, this catalogue consisting on various leaflets (TOSCA Documents) is available in singular format with unified accessibility interface. Broker is divided into three logical blocks depending upon its functional scope. The first blocks with dotted line on top is service provider, followed by core TSB marked in solid line and ending with users in dotted line at the bottom. The workflow is from Provider to Users with TSB in between. This pivotal placement of TSB is a key to address the challenges of service consumers.

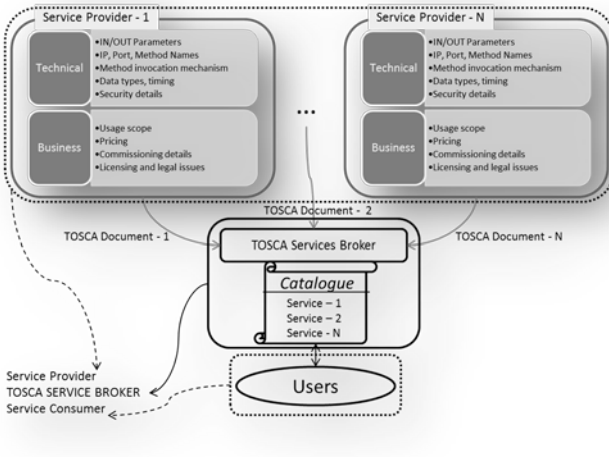

Fig. 3. TOSCA Service Broker Workflow

The crucial component of TSB is its DB schema (Figure 4.) which holds metadata derived from TOSCA definitions. Three tables are designed to hold the details of services that include 'Service Provider', 'Service Usage' and 'Service Types'. Primary key is placed into 'Service Provider' which is further mapped as a secondary key in rest of the tables. This key is unique index for each service provider. Index

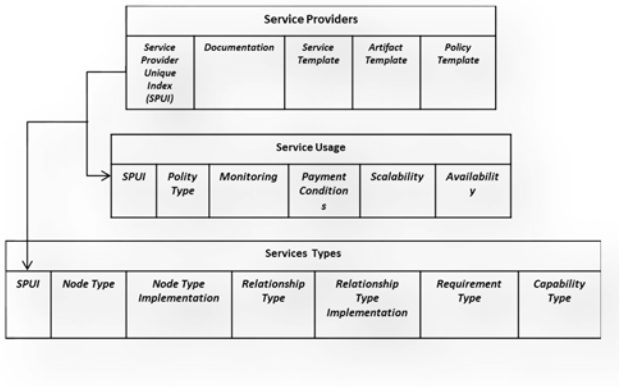

Fig. 4. TOSCA Service Broker Database Schema

represents documentation, service, artifact and policy templates. 1) Service Templates are used to deploy the application on cloud platforms. A single service template can define an application or it can be used to compose service form other applications. The structure of Cloud applications is described in Service template that defines plans to manage the offered services. Node templates and node types can be defined in the Service template scope which actually presents offered service. 2) Artifact Template describes application payload and its software components. It is presented as a compressed data including binaries, scripts and static files. Payload can be part of artifact or it can be fetched from web. And finally, 3) Policy Template holds details of polices that are non-functional behavioural properties of provided service and stored in 'Service Usage' table. These include Monitoring, Payment, Conditions, Scalability and Availability.

TSB Design:

TSB is designed on modular basis where set of functionality are dedicated to modules as presented in Figure 5. The modularized approach is adopted to achieve software extensibility and efficient debugging. Beside advantages of modularized software development, the proposed architecture manages the implementation with logical construction of the TSB as below:

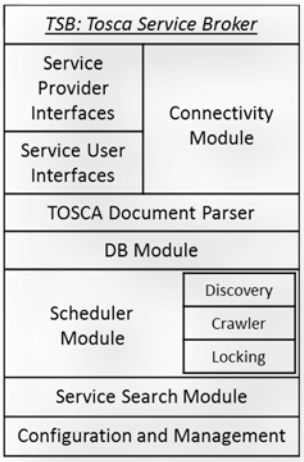

Fig. 5. TOSCA Service Broker Design – Breakdown into modules with respect to functionality.

A. *Connectivity Module:*

Connectivity Module is designed to meet the requirements of different service providers and users. It provides separation between the connectivity and functionally modules such as TOSCA Document Parser, DB Module and Scheduler Module. The Connectivity Module delivers special features which limit control of certain networking functions from the module itself. The module supports both Client and Server connectivity for service providers and users interfaces respectively.

B. *TOSCA Document Parser:*

Document parser is relatively passive component as it relies on connectivity module to fetch descriptions from sources and DB Module to make commits. This module defaults to XML format which is standard to TOSCA, however, any other text based formats such as JASON/YAML etc. can be added without any architectural changes.

C. DB Module:

The DB Module enables XML Parser and User Interface modules to connect with database. This module consists of stored procedures and also allows executing custom database quires including Select, Insert, Update or Delete. All the metadata parsed from documents and stored in DB is only accessed via this module.

D. Scheduler Module:

Scheduler Module is responsible for synchronization, event handling and scheduling various tasks. It is further divided into three sub-modules.

1) Discovery: sub-module is responsible to listen to available services that comply with WSDL. Once the WSDL discovery event is raised then Discovery component will initialize service provider interface and document parser module to update the database for that specific provider.
2) Crawler: service discovery depends upon implementation and it is not a mandatory even in TOSCA/WSDL. A service crawler is a component that would systematically search for service providers for offered service and update or add service index that do not implement discovery mechanism.
3) Locking: sub-module ensures that there are no deadlocks in the broker. If discovery or crawler is updating database then service search module or parser module should wait unless the updates are committed. It provides system level locks for each module.

E. Service Search Module:

This module plays vital role in the broker to provide best service from the catalogue. It accepts user input criteria and scans the catalogue's leaflets for most efficient solution in terms of usage and price. It is important to note that various services can exit with similar functionality, so it is vital role of Search Module to analytically search the best solution.

F. Configuration and Management:

Configuration and Management module provides various supporting functionality to all other modules. For example, timer settings for crawler, DB settings, Parser keys, service search algorithm. It also provides system level auditing, logging, performances details etc.

IV. Future Work

Proposed TSB strictly relies on mandatory metadata fields in TOSCA document and makes use of fewer optional elements. Although, optional elements are function of implementation but TSB should extend DB schema to accommodate these fields. These optional elements - provided in TOSCA document - can increase the throughput of Service Search Module. However, Service Search Module itself requires improved examination capability other than simple match and report mechanism against user input. The consequences of such a short coming can lead to incorrect selection in presence of more capable and cost-effective service. Components of Scheduler Module depend on Configuration and Management Module which results in dependency on human interaction. For example, discovery component or crawler has no means to keep the record and update the previous status of existing services in the system.

V. Conclusion

Cloud services are omnipresent in these days. The standardization to describe these services is focused on deployment and management. This is leading to challenges for service consumers to select the most competent solution. Current solution is to utilize regular web search engines and individually compare all the detail. Manual search and selection is not only time consuming but also makes it impossible to automate user application. This lack of automation - results to source level changes in applications to find superior services. A service broker not only provides ability to automate user application but also to select the most cost effective service providers with greater QoS.

On lines of economics in services echo-system, middleman design would ensure fair competition and encourage service provider to describe their packages in lengths and compare with benchmarks and market trends. Service providers can customize their services as per market trends and quickly disperse updates across huge consumer base without updating every single consumer.

References

[1] J. Cardoso, T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "Cloud computing automation: Integrating usdl and tosca," in *Advanced Information Systems Engineering*, 2013, pp. 1–16.

[2] OASIS Committee, "Topology and Orchestration Specification for Cloud Applications (TOSCA)--Committee Specification 01," 2013.

[3] S. Qanbari, F. Li, and S. Dustdar, "Toward Portable Cloud

Manufacturing Services," *Internet Comput. IEEE*, vol. 18, no. 6, pp. 77–80, Nov. 2014.

[4] M. Guzek, A. Gniewek, P. Bouvry, J. Musial, and J. Blazewicz, "Cloud Brokering: Current Practices and Upcoming Challenges," *Cloud Comput. IEEE*, vol. 2, no. 2, pp. 40–47, Mar. 2015.

[5] M. Ghijsen, J. der Ham, P. Grosso, and C. De Laat, "Towards an infrastructure description language for modeling computing infrastructures," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, 2012, pp. 207–214.

[6] T. Baker, A. Hussien, M. Randles, and A. Taleb-Bendiab, "Supporting elastic cloud computation with intention description language," in *PGNet2010: The 11th Annual Conference on the Convergence of Telecommunications, Networking & Broadcasting*, 2010.

[7] T. Binz, G. Breiter, F. Leyman, and T. Spatzier, "Portable Cloud Services Using TOSCA," *Internet Comput. IEEE*, vol. 16, no. 3, pp. 80–85, May 2012.

[8] L. Sun, H. Dong, and J. Ashraf, "Survey of service description languages and their issues in cloud computing," in *Semantics, Knowledge and Grids (SKG), 2012 Eighth International Conference on*, 2012, pp. 128–135.

[9] J. Cardoso, A. Barros, N. May, and U. Kylau, "Towards a unified service description language for the internet of services: Requirements and first developments," in *Services Computing (SCC), 2010 IEEE International Conference on*, 2010, pp. 602–609.

[10] A. Charfi, B. Schmeling, F. Novelli, H. Witteborg, and U. Kylau, "An overview of the unified service description language," in *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, 2010, pp. 173–180.

[11] A. Brogi, J. Soldani, and P. Wang, "TOSCA in a Nutshell: Promises and perspectives," in *Service-Oriented and Cloud Computing*, Springer, 2014, pp. 171–186.

[12] G. Gonçalves, P. Endo, M. Santos, D. Sadok, J. Kelner, B. Melander, and J.-E. Mångs, "Cloudml: An integrated language for resource, service and request description for d-clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 2011, pp. 399–406.

[13] G. Schaffrath, S. Schmid, I. Vaishnavi, A. Khan, and A. Feldmann, "A resource description language with vagueness support for multi-provider cloud networks," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, 2012, pp. 1–7.

[14] G. C. Silva, L. M. Rose, and R. Calinescu, "Cloud DSL: A Language for Supporting Cloud Portability by Describing Cloud Entities," *CloudMDE 2014*, p. 36, 2014.

[15] J. Cardoso and C. Pedrinaci, "Evolution and overview of Linked USDL," in *Exploring Services Science*, Springer, 2015, pp. 50–64.

[16] C. Pedrinaci, J. Cardoso, and T. Leidig, "Linked USDL: a vocabulary for web-scale service trading," in *The Semantic Web: Trends and Challenges*, Springer, 2014, pp. 68–82.

[17] J. George, F. Belqasmi, R. H. Glitho, and N. Kara, "A Substrate Description Framework and Semantic Repository for Publication and Discovery in Cloud Based Conferencing.," in *CSWS*, 2013, pp. 41–44.

[18] A. Lenk, C. Dänschel, M. Klems, D. Bermbach, and T. Kurze, "Requirements for an IaaS deployment language in federated Clouds," in *Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on*, 2011, pp. 1–4.

[19] P. Hoberg, J. Wollersheim, and H. Krcmar, "Service Descriptions for Cloud Services-The Customer's Perspective," in *Proceedings of ConLife Academic Conference*, 2012.